日本ソフトウェア科学会第 28 回大会 (2011 年度) 講演論文集

VM マイグレーションを可能にする IDS オフロード機構

宇都宮 寿仁　光来 健一

本稿では、仮想マシン（VM）のマイグレーションを行う際に、オフロードした侵入検知システム（IDS）が監視を継続できるようにする新しいオフロード機構を提案する。IDS はサーバへの不正アクセスを検出するために用いられているが、それ自身が先に攻撃されるとセキュリティが低下してしまう。そこで、サーバと IDS を別々の仮想マシンで動かして安全に監視する IDS オフロードと呼ばれる手法が提案されている。しかし、サーバマシンのメンテナンスなどのために仮想マシンを別のホストにマイグレーションすると、オフロードした IDS は監視を継続できなくなる。我々は Xen にドメイン M と呼ばれるオフロード専用仮想マシンを導入し、マイグレーション後もストレージやメモリの監視を継続できるようにした。

1 はじめに

サーバへの不正アクセスが年々増加してきている。このような攻撃を検出するために侵入検知システム (IDS) が用いられている。IDS はストレージやネットワーク、メモリの内容などの監視を行い、攻撃を検出した際には管理者に通知する。例えば、Tripwire [8] はファイルの改ざんを検出することができ、Snort [12] は不正なネットワークパケットを検出するために用いられている。しかし最近は、まず IDS を攻撃して停止させてからサーバへの攻撃を行うことが増えてきている [6]。この場合、IDS が攻撃を検知することはできなくなる。

IDS への攻撃を緩和する方法として仮想マシンを用いた IDS オフロードという手法が提案されている [5] [7] [9]。この手法は監視対象のサービスとそれを監視する IDS を別々の仮想マシンで動作させることで IDS の安全な実行を可能にする。IDS オフロードを行った際に問題になるのが、仮想マシンのマイグレーションである。IDS をオフロードした先の仮想マシンは監視対象の仮想マシンと一緒に別のホストにマイグレーションすることができないため、IDS による監視を継続することができなくなる。

本稿では、オフロードされた IDS を動作させることができ、マイグレーションも行うことができるオフロード専用仮想マシンであるドメイン M を提案する。ドメイン M で動く IDS は監視対象の仮想マシンのストレージやメモリを監視することができる。また、ドメイン M 上の IDS が監視を行っている最中に、監視対象の仮想マシンと一緒にマイグレーションを行うことができる。マイグレーション先では IDS の監視を再開し、途中の状態から監視を継続することができる。

我々は Xen [2] を用いてドメイン M を実装した。ドメイン M からストレージを監視できるようにするために、監視対象の仮想マシンのストレージには NFS サーバを用いる。メモリの監視を行うためにドメイン M に特権を与え、マイグレーション時にメモリページのマップ状態を保つためにページテーブルの操作を行う。ドメイン M で Tripwire を動かしてストレージを監視し、カーネルのテキスト領域のハッシュ値を計算することでカーネルメモリを監視する実験を行った。その結果、正常に監視を行うことができ、マイグ

A New IDS Offload Mechanism Enabling VM Migration.

Hisato Utsunomiya, Kenichi Kourai, 九州工業大学, Kyushu Institute of Technology.

レーション後も監視を継続できることが確認できた。

以下、2 章では IDS オフロードを行った際のマイグレーションの問題点について述べる。3 章でドメイン M について述べ、4 章でドメイン M 上で IDS を動作させた実験について述べる。5 章で関連研究に触れ、6 章で本稿をまとめる。

2 IDS オフロード時のマイグレーション

2.1 IDS オフロード

仮想マシンを用いた IDS のオフロードは、攻撃を受ける可能性のあるサービスを仮想マシン上で動作させ、IDS だけを別の仮想マシンで動かす手法である。オフロードされた IDS はオフロード元の仮想マシンのストレージやネットワーク、メモリ上の OS のデータなどを安全に監視することができる。IDS をオフロードすることにより、オフロード元の仮想マシンが攻撃を受けたとしても、IDS は影響を受けない。オフロード元の仮想マシンからオフロード先の仮想マシンへのアクセスはできないためである。オフロード先の仮想マシンは外部にサービスを提供する必要がないため、攻撃を受ける可能性は低い。

仮想化ソフトウェアの Xen を用いた IDS オフロードでは、図 1 のように、サービスはドメイン U と呼ばれる仮想マシンで動作させ、IDS はドメイン 0 と呼ばれる仮想マシンで動作させる。ドメイン U は通常の仮想マシンであり、ドメイン 0 はホスト上に一つしか存在せず、ドメイン U の管理を行う特権を持った仮想マシンである。Xen においてはドメイン 0 がドメイン U にストレージやネットワークを提供しているため、ドメイン 0 で動作する IDS はこれらを容易に監視することができる。ドメイン U のメモリに関しても、仮想マシンモニタを介してアクセスすることができる。

2.2 マイグレーション

仮想マシンの特徴の一つにマイグレーションと呼ばれる操作があり、仮想マシンをあるホストから別のホストに移動させることができる。この際に仮想マシンの実行状態を保ったまま移動させるため、マイグレーション先のホストですぐに実行を再開することができる。例えば、物理マシンのメンテナンスを行う際にマイグレーションが用いられる。メンテナンス時には物理マシンを停止させなければならないため、仮想マシン上のサービスも停止してしまう。メンテナンス前に仮想マシンを別のホストにマイグレーションしておくことによって、サービスを継続したまま物理マシンのメンテナンスを行うことができる。

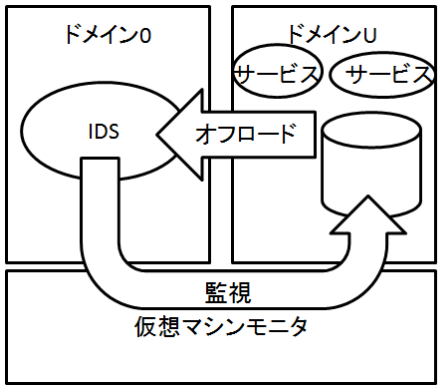

図 1 IDS オフロード

マイグレーションは大きく分けて以下の 3 ステップで行われる。マイグレーション元のホストで仮想マシンの実行を停止しサスペンドを行う。サスペンドは仮想マシンのメモリの内容や実行状態を保存する操作である。次に、マイグレーション元からマイグレーション先のホストににサスペンドした仮想マシンのイメージを転送する。その後、受信したイメージを元にマイグレーション先で仮想マシンをレジュームして再開する。レジュームはサスペンドされた仮想マシンを復元する操作である。

2.3 問題点

Xen において IDS オフロードを行った場合、仮想マシンのマイグレーションを正しく行うことができなくなる。マイグレーション後にも IDS による監視を継続するには、ドメイン U とドメイン 0 を両方ともマイグレーションする必要がある。しかし、IDS をオフロードしたドメイン 0 はマイグレーションすることができない。ドメイン 0 はドメイン U のストレージやネットワークへのアクセスを中継しており、物理デバイスに強く依存しているため、サスペンドを行う

のが難しい。また、ドメイン 0 はホストに 1 つだけ存在する特殊な仮想マシンであるため、マイグレーションするとそのホストにはドメイン 0 が存在しなくなってしまう。このように、ドメイン U だけしかマイグレーションできできないため、IDS による監視を継続できなくなる。

この問題への解決策として、ドメイン U のマイグレーションと同時にマイグレーション先のドメイン 0 で IDS を実行し直して監視を行うという手法が考えられる。IDS の設定ファイルやデータベース等をマイグレーション先と共有することで、マイグレーション前と同様にドメイン U の監視を行うことができる。しかし、IDS の監視の途中から再開させることは難しい。例えば、Tripwire によるストレージ全体の検査には時間がかかるが、検査中にドメイン U をマイグレーションすると、マイグレーション先で最初から検査し直すことになる。

3 ドメイン M

本稿では、オフロードした IDS を動作させることができ、別ホストへの VM マイグレーションが可能であるオフロード専用仮想マシンを提案する。この仮想マシンはドメイン M と呼ばれる。ドメイン M はドメイン 0 と同様に、ドメイン U のストレージやメモリの監視を行うことができる。また、ドメイン M は監視対象のドメイン U と一緒にマイグレーションを行うことができ、マイグレーション後も監視を継続することができる。

3.1 ストレージの監視

ドメイン M はドメイン 0 と同様に、ドメイン U の仮想ディスクイメージを参照することでストレージの監視を行う。ドメイン U の仮想ディスクイメージはドメイン 0 に置かれるため、ドメイン 0 ではこのイメージファイルをループバックマウントすることができる。この際に、ドメイン U で使われているファイルシステムを指定することで、ファイルやディレクトリを認識することができる。しかし、ドメイン M からは同様の手法でイメージファイルをマウントすることはできない。

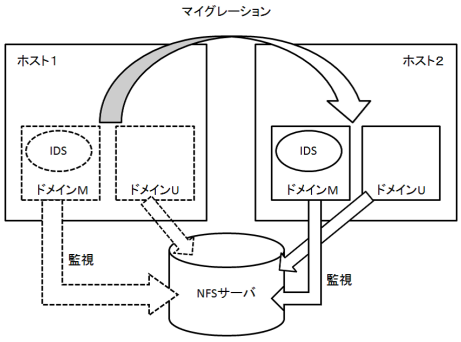

図 2 マイグレーション後のストレージの監視

ドメイン M がドメイン U のストレージを監視できるようにするために、図 2 のように NFS サーバを用いる。NFS サーバ上にドメイン U 用の仮想ディスクイメージを配置しておき、ドメイン M はこのディスクイメージを含む領域を NFS マウントする。これにより、ドメイン U のストレージを監視することができる。一方、このディスクイメージを含む領域をドメイン 0 にもマウントし、そのディスクイメージを使ってドメイン U を起動する。ドメイン U をマイグレーションできるようにするためにも、NFS サーバ上に仮想ディスクイメージを配置することが必要である。

NFS サーバを用いることで、ドメイン M のマイグレーション後もドメイン U のストレージの監視を継続することができる。ドメイン M はマイグレーション後も IP アドレスが変わらないため、NFS マウント状態を保持することができる。また、ドメイン M をマイグレーションできるようにするために、ドメイン M の仮想ディスクイメージも NFS サーバ上に配置する。

3.2 メモリの監視

3.2.1 Xen のメモリモデル

Xen ではマシン全体の物理メモリはマシンメモリと呼ばれ、各ドメインに割り当てられる物理メモリは疑似物理メモリと呼ばれる。マシンメモリにはマシンフレーム番号 (MFN) と呼ばれる番号が 0 から順番に付けられており、疑似物理メモリには疑似物理フレーム番号 (PFN) と呼ばれる番号が 0 から順番に付けられている。今回対象としている準仮想化環境では各

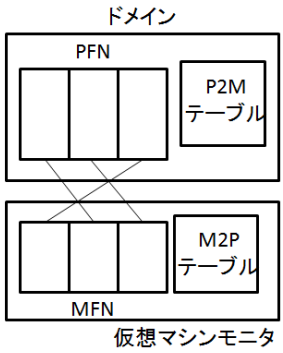

図 3　Xen のメモリ管理

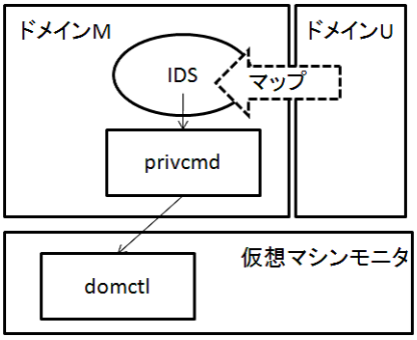

図 4　ドメイン M におけるメモリマップ

ドメインは図 3 のような PFN から MFN への対応表を保持しており、この対応表は P2M テーブルと呼ばれる。一方、MFN から PFN への対応表は仮想マシンモニタが保持しており、M2P テーブルと呼ばれる。各ドメインでは仮想アドレスに対応する物理メモリフレームをページテーブルに登録する際に、PFN ではなく MFN を用いる。この際に P2M テーブルを用いて PFN を MFN に変換する。これにより MMU が仮想アドレスを MFN に直接変換することができる。

3.2.2　ドメイン U のメモリマップ

ドメイン M はドメイン 0 と同様に、ドメイン U のメモリページをマップすることでメモリの監視を行う。ドメイン M はドメイン U のメモリ上のページテーブルを参照して仮想アドレスを MFN に変換する。ページテーブルの参照の際にもドメイン U のメモリページをマップする必要がある。最終的に得られた MFN の指すメモリページをマップすることで目的のメモリの監視を行う。

ドメイン M では Xen のスタブドメインの機能を利用して監視先のドメイン U のメモリへのアクセス権を与える。従来、ドメイン U のメモリを参照できるのはドメイン 0 のみであった。スタブドメインは指定したドメイン U のメモリにアクセスすることができる。ドメイン M に特定のドメイン U へのアクセス権を与えるには、図 4 のようにドメイン 0 から domctl ハイパーコールを用いて設定する。

さらに、ドメイン U のメモリページをマップできるように、ドメイン M の Linux カーネルにサポートを追加する。ドメイン 0 は privcmd というインタフェースを用いてドメイン U のメモリページのマップを行う。しかし、ドメイン 0 以外に privcmd インタフェースが存在しないため、ドメイン M にも追加した。また、カーネル内で privcmd の利用をドメイン 0 にだけ許可しているため、ドメイン M にも許可するようにした。

加えて、ドメイン U のメモリ情報を取得するために呼び出す domctl ハイパーコールの実行をドメイン M にも許可する。このハイパーコールはドメイン U を管理するために使われるため、ほとんどの機能はドメイン 0 にしか許可されていない。そこでドメイン M も domctl ハイパーコールの実行を行えるようにした。

3.2.3　マイグレーション時のメモリ監視の継続

ドメイン M をマイグレーションできるようにするには、ドメイン U のメモリページのマップ状態をマイグレーション後に復元できるようにする必要がある。マップ状態が復元できなければ、マイグレーション後にメモリの監視を継続することができなくなる。しかし、従来、マイグレーション可能だったドメイン U では他のドメイン U のメモリをマップすることはできなかったため、この点については考慮されていない。その結果、マイグレーションの最初の段階のサスペンド処理に失敗する。

そこで、ドメイン M のサスペンドを行う際に、ドメイン U のメモリページのマップ状態についても保存を行う。準仮想化環境では図 5 のように、ドメインのサスペンド処理の際に M2P テーブルを用いて、ペー

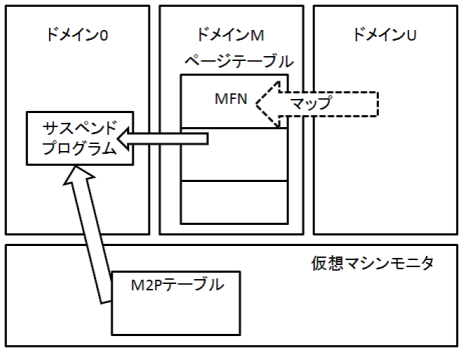

図 5　サスペンド時の動き

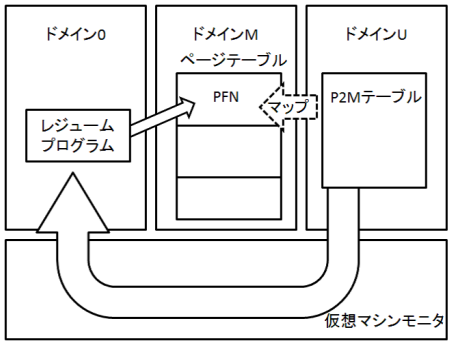

図 6　レジューム時の動き

ジテーブルエントリの中の MFN を対応する PFN に変換する。これはホストに依存しないメモリ参照である PFN を用いることで、仮想マシンを別のホストに移動させられるようにするためである。ドメイン M ではこの変換の際に、MFN がドメイン U のメモリページを指している場合ドメイン U 内の対応する PFN に変換する。ドメイン M の PFN とドメイン U の PFN を区別するために、ページテーブルエントリに DomU ビットと呼ばれるビットを追加した。ページテーブルエントリがドメイン U のメモリページを指している場合にはこの DomU ビットを 1 にセットする。

サスペンドしたドメイン M をマイグレーション先ホストでレジュームする際にはドメイン U のメモリページのマップ状態についても復元を行う。ドメイン M のレジューム処理では、サスペンド処理とは逆に、図 6 のように P2M テーブルを用いてページテーブルエントリの中の PFN を対応する MFN に変換する。この際に、ページテーブルエントリの domU ビットが 1 であれば、監視しているドメイン U の PFN に対応する MFN に変換する。この変換にはドメイン U の P2M テーブルが必要になるため、ドメイン U のメモリ上に置かれた P2M テーブルをメモリマップすることにより参照する。

このように復元したページテーブルを有効化できるようにするために、ドメイン 0 がドメイン U のメモリページをドメイン M にマップすることを可能にした。レジューム処理はドメイン 0 によって行われるため、ドメイン 0 からドメイン M とドメイン U の間のメモリマップの設定を行えるようにする必要がある。しかし、従来の仮想マシンモニタではこのようなメモリマップを想定していなかったため、ドメイン M のページテーブルの登録に失敗していた。そこで、ページテーブルの登録を行うハイパーコール内のチェックを修正することで、ドメイン 0 からのメモリマップをの設定を行えるようにした。

マイグレーション先に監視対象のドメイン U の情報を送るには、ドメイン M のコンフィグに追加した target_uuid オプションを用いる。このオプションに監視先のドメイン U の UUID を設定してドメイン M を起動する。このコンフィグはドメイン M のマイグレーション時に一緒に送られ、マイグレーション先ではコンフィグ内の UUID を基に監視対象のドメイン U を見つける。現在の実装では、ドメイン M より先にドメイン U のマイグレーションが完了していることを仮定している。

4　実験

我々はドメイン M を Xen 4.0.1 に実装し、ドメイン M にオフロードした IDS の性能、および、ドメイン M のマイグレーションのオーバヘッドを調べる実験を行った。実験にはマイグレーション元とマイグレーション先として、Intel Core 2 Quad 2.83GHz の CPU 、4GB のメモリ、ギガビットイーサネットを搭載した PC を 2 台用いた。 ドメイン 0、ドメイン U、ドメイン M ともに Linux カーネル 2.6.32.25 を動作させ、それぞれ 4GB、512MB、512MB のメモリを割り当てた。NFS サーバには、Intel Xeon X5640

表 1 **Tripwire** の実行時間

	時間 (秒)
ドメイン 0 から監視	4.5
ドメイン M から監視	18.9

表 2 ストレージ監視の時間

	時間 (秒)
NFS マウントなし	92.6
NFS マウントあり	92.3

3.16GHz の CPU、32GB のメモリ、SATA 16TB の RAID5 構成、ギガビットイーサネット、Open-E OS を搭載した NAS を用いた。

4.1 ストレージの監視

ドメイン M 上で Tripwire を動作させ、ドメイン U のストレージを監視する実験を行った。Tripwire はドメイン M 上で正常に動作し、ドメイン U のストレージの改ざんを検出できることを確認した。ストレージ監視の性能を調べるためにドメイン M から監視した場合とドメイン 0 から監視した場合を比較した。10 回計測した平均値を表 1 に示す。この結果より、ドメイン 0 から監視を行ったほうが実行時間が短いことが分かった。これはドメイン M が NFS サーバにアクセスする場合、そのネットワークアクセスがドメイン 0 経由となってしまうためである。ネットワーク仮想化のオーバヘッドの分だけドメイン 0 から監視するより低速になったと考えられる。

次に、Tripwire で監視を行っている状態でドメイン M のマイグレーションを行った。マイグレーション後もドメイン M 上の Tripwire はドメイン U の監視を継続していることを確認した。また、ドメイン U のストレージの NFS マウントの有無がマイグレーション時間に与える影響を調べた。マイグレーション時間を 10 回計測した平均値を表 2 に示す。NFS マウントの有無に関わらず、マイグレーション時間はほぼ同じであった。NFS マウントを行っている方がわずかに高速であるが、誤差の範囲だと考えられる。

4.2 メモリの監視

ドメイン M で OS カーネルのテキスト領域のハッシュ値を SHA-1 を用いて計算する IDS を動作させ、ドメイン U のカーネルメモリを監視する実験を行った。この IDS はドメイン M 上で正常に動作し、ドメイン U のカーネルの改ざんを検出できることを確認した。メモリ監視の性能を調べるために、ドメイン M から監視した場合とドメイン 0 から監視した場合の実行時間を比較した。ドメイン M がマップしたドメイン U のメモリページは 1448 ページであった。カーネルのテキスト領域のハッシュ値の計算を 100 回繰り返して 1 回あたりの計算時間を求め、それを 10 回計測した平均値を表 3 に示す。この結果より、ドメイン M から監視を行ったほうが実行時間が短いことが分かった。ドメイン 0 とドメイン M とでメモリマップの処理は同じであるため、ドメイン M の方が高速である原因の調査については今後の課題である。

表 3 カーネルメモリを監視する **IDS** の実行時間

	時間 (ミリ秒)
ドメイン 0 から監視	239
ドメイン M から監視	135

表 4 メモリ監視のマイグレーション時間への影響

	時間 (秒)
メモリマップなし	92.6
メモリマップあり	92.2

次に、この IDS で監視を行っている状態でドメイン U とドメイン M のマイグレーションを行った。その結果、マイグレーション後もドメイン U の監視を継続することができていた。さらに、ドメイン M がドメイン U のメモリをマップしている状態と、マップしていない状態でそれぞれマイグレーションを行い、その時間を計測した。10 回計測した平均値を表 4 に示す。メモリのマップ状態に関わらず、マイグレーション時間はほぼ同じであった。

5 関連研究

オフロードした IDS が動いている仮想マシンをマイグレーションする代わりに、IDS だけをプロセスマイグレーションする方法も考えられる。プロセスマイグレーションは VM マイグレーションと似ており、プロセスをホスト間で移動させることができる。実行中のプロセスの状態を保存して別のホストに転送し、実行を再開する。ストレージについては、NFS サーバ上に置いておくことでプロセスマイグレーション後もアクセスすることができる。しかし、他の仮想マシンのメモリマップ状態はプロセスマイグレーションによって失われるため、ドメイン M と同様にしてメモリマップ状態の復元を行う必要がある。また、MOSIX [1] や BLCR [3] などは OS のカーネルを大幅に変更する必要があり、Condor [10] や libckpt [11] などは IDS の修正を必要とする。ドメイン M が必要とするカーネルへの変更はごくわずかであり、IDS の修正は不要である。

Xen のスタブドメインはドメイン 0 の一部の機能をドメイン 0 とは別の仮想マシンで動かすために用いられる。例えば、完全仮想化環境で I/O エミュレーションを行う qemu をスタブドメインで動作させることができる。スタブドメインでは一般的に Mini-OS と呼ばれる最小構成のゲスト OS が用いられるが、Mini-OS はシングルスレッド OS であるため、既存のソフトウェアをそのまま動かせないことが多い。ドメイン M はスタブドメインをベースとしているが、スタブドメインの機能を利用できるように修正した Linux を動かす。そのため、既存のソフトウェアをそのまま動かすことができる。

ドライバドメイン [4] はディスクやネットワークのデバイスドライバをドメイン 0 とは別の仮想マシンで動かすために用いられる。ドメイン U はドメイン 0 の代わりにドライバドメインを経由してディスクやネットワークにアクセスすることができる。そのため、ドライバドメインで IDS を動かし、ドメイン U のディスクやネットワークを監視することも可能である。しかし、ドライバドメインも物理デバイスに強く依存しているため、マイグレーションを行うのは難しい。

Xen 以外で IDS オフロードを行うシステムも提案されているが、IDS まで含めたマイグレーションを考慮しているシステムはない。Livewire [5] や VMwatcher [7] はホスト OS 上に IDS をオフロードして仮想マシンの監視を行う。ホスト OS にオフロードした場合、IDS は仮想マシン上で動かないためプロセスマイグレーションを行う必要がある。HyperSpecor [9] は OS 上に独立した仮想環境を作り、そこに IDS をオフロードする。OS レベルでの仮想化を利用しているため、この仮想環境のマイグレーションを行うのは難しい。

6 まとめと今後の課題

本稿ではマイグレーション後も監視を継続することができる IDS オフロード機構であるドメイン M を提案した。ドメイン M には IDS をオフロードすることができ、安全にドメイン U の監視を行うことができる。また、ドメイン M に IDS オフロードを行った状態でマイグレーションすることができ、マイグレーション先のホストで IDS がドメイン U の監視を途中から再開することができる。

今後の課題として、ドメイン M からドメイン U のネットワークの監視を行うことが挙げられる。また、ドメイン M を停止させずに移動させるライブマイグレーションへの対応も行う必要がある。

謝辞

本研究の一部は、科学技術振興機構 戦略的創造研究推進事業 (CREST) 研究領域「実用化を目指した組み込みシステム用ディペンダブル・オペレーティングシステム」による。

参 考 文 献

[1] Barak, A. and La'adan, O.: The MOSIX multicomputer operating system for high performance cluster computing, *Future Generation Computer Systems*, Vol. 13, No. 4-5(1998), pp. 361–372.

[2] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A.: Xen and the Art of Virtualization, *Proc. Symp. Operating Systems Principles*, 2003, pp. 164–177.

[3] Duell, J.: The design and implementation of

Berkeley Lab's Linux Checkpoint/Restart, Technical report, Lawrence Berkeley National Laboratory, 2003.

[4] Fraser, K., Hand, S., Neugebauer, R., Pratt, I., Warfield, A., and Williamson, M.: Safe Hardware Access with the Xen Virtual Machine Monitor, *Proc. Workshop on Operating System and Architectural Support for the on demand IT InfraStructure*, Oct 2004.

[5] Garfinkel, T. and Rosenblum, M.: A Virtual Machine Introspection Based Architecture for Intrusion Detection, *Proc. Network and Distributed Systems Security Symp.*, 2003, pp. 191–206.

[6] IBM: Stick - A Protential Denial of Service Against IDS Systems, *Internet Security Systems Security Alert*, 2001.

[7] Jiang, X., Wang, X., and Xu, D.: Stealthy Malware Detection through VMM-based "Out-of-the-box" Semantic View Reconstruction, *Proc. Conf. Computer and Communications Security*, 2007, pp. 128–138.

[8] Kim, G. and Spafford, E.: The Design and Implementation of Tripwire and A File System Integrity Checker, *Proc. Conf. Computer and Communications Security*, 1994, pp. 18–29.

[9] Kourai, K. and Chiba, S.: HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection, *Proc. Int. Conf. Virtual Execution Environments*, 2005, pp. 197–207.

[10] Litzkow, M., Tannenbaum, T., Basney, J., and Livny, M.: Checkpoint and migration of UNIX processes in the Condor distributed processing system, Technical report, University of Wisconsin-Madison Computer Sciences Department, Apr 1997.

[11] Plank, J. S., Beck, M., Kingsley, G., and Li, K.: Libckpt: Transparent checkpointing under Unix, *Proc. Usenix Winter Technical Conference*, Jan 1995, pp. 213–223.

[12] Roesch, M.: Snort–Lightweight Intrusion Detection for Networks, *Proc. USENIX System Administration*, 1999.